AMT630HV160 启动说明

版本：V1.0

日期：2024-11

深圳开阳电子股份有限公司拥有随时修改本手册的权利，内容如有更改，恕不另行通知。深圳开阳电子股份有限公司对本手册不承担任何形式的保证，包括但不限于对产品特定用途适销性和适用性的隐含保证。深圳开阳电子股份有限公司对本手册中包含的错误或对本手册的使用所带来的偶然或继起损害不承担任何责任。

版本记录：

日期	版本号	更新说明	
2024-11	V1.0	初始建立版本	

1. 启动流程

1.1 安全启动的作用

客户通常会有不允许在机器上运行未授权的软件的需求，而安全启动程序提供了这种保证，因为 BOOTROM 在芯片重启后总是最先执行且不能被修改。当启用安全启动时，BOOTROM 会检查加载到芯片 RAM 中的第一个执行镜像，以确定该代码的安全性(RSA2048 PKCS1-v1_5 认证)。如果代码是安全的，那么程序执行将转移到它身上，这样可以建立一个从 BOOTROM 到用户程序的受信任代码链。

可能还有些客户会有保证代码机密性的需求，用户可以使用密钥对程序镜像进行对称加密(AES-GCM)，而 BOOTROM 可以使用相同的密钥对加密后的镜像进行完整性验证以及解密后再执行。(加解密使用的密钥需要烧录到 EFUSE 的 0 地址，该密钥内容除了安全模块本身其他模块均不能访问)

1.2 功能介绍

64KB 的片上引导 ROM，便于将镜像加载到片上 SRAM 中，并支持以下功能：

- 支持使用 RSASSA-PKCS1-v1_5 签名验证与 2048 位公钥（2048 位模量，32 位指数）进行引导镜像验证。
- 支持 Root of Trust key 的使用，通过比较 RoT 公钥的哈希值和 EFUSE 保存的值是否一致来验证镜像密钥证书。
- 支持 AES- GCM 加密镜像。
- 支持以下安全引导镜像。
  - RSA-2048 签名镜像。
  - AES-GCM（128 或 256 位）加密镜像。
  - 加密后再签名镜像或者是签名后再加密镜像。

1.3 启动流程

片上 ROM 用于存储安全引导代码。每次设备开机或重启时，都会执行引导加载程序代码。启动流程大致分为以下部分：

1. 判断启动源是从管脚读取启动介质还是直接从介质启动(详见图 1.1)。
2. 安全或者非安全介质启动(详见图 1.2)。

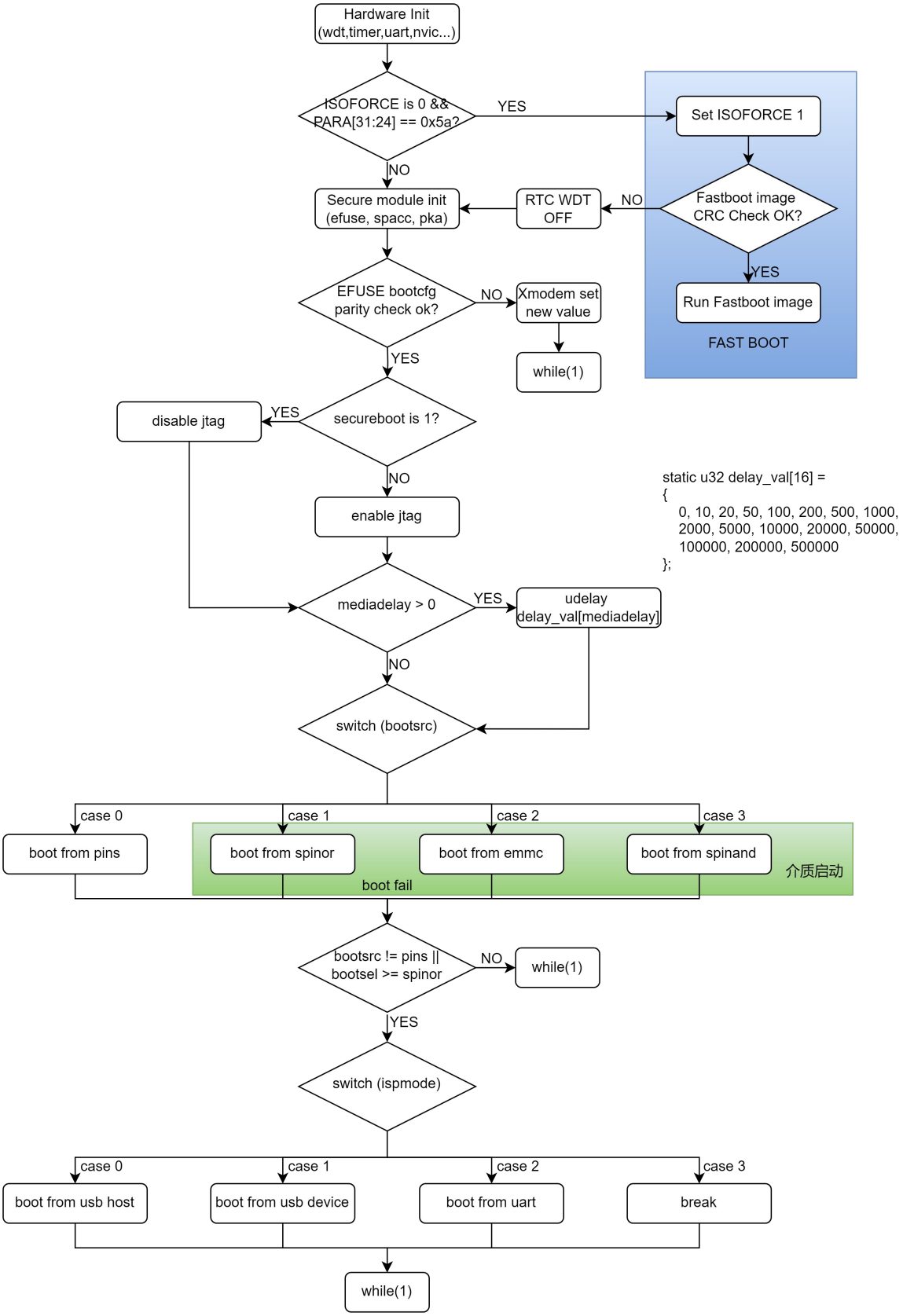
Hardware Init
(wdt,timer,uart,nvic...)
ISOFORCE is 0 &&
PARA[31:24] == 0x5a?
YES
Set ISOFORCE 1
NO
Fastboot image
CRC Check OK?
NO
RTC WDT
OFF
Secure module init
(efuse, spacc, pka)
YES
Run Fastboot image
FAST BOOT
EFUSE bootcfg
parity check ok?
NO
Xmodem set
new value
while(1)
YES
disable jtag
YES
secureboot is 1?
NO
enable jtag
static u32 delay_val[16] =
{
0, 10, 20, 50, 100, 200, 500, 1000,
2000, 5000, 10000, 20000, 50000,
100000, 200000, 500000
};
mediadelay > 0
YES
udelay
delay_val[mediadelay]
NO
switch (bootsrc)
case 0
case 1
case 2
case 3
boot from pins
boot from spinor
boot from emmc
boot from spinand
介质启动
boot fail
bootsrc != pins ||
bootsel >= spinor
NO
while(1)
YES
switch (ispmode)
case 0
case 1
case 2
case 3
boot from usb host
boot from usb device
boot from uart
break
while(1)

图 1.1 启动流程图

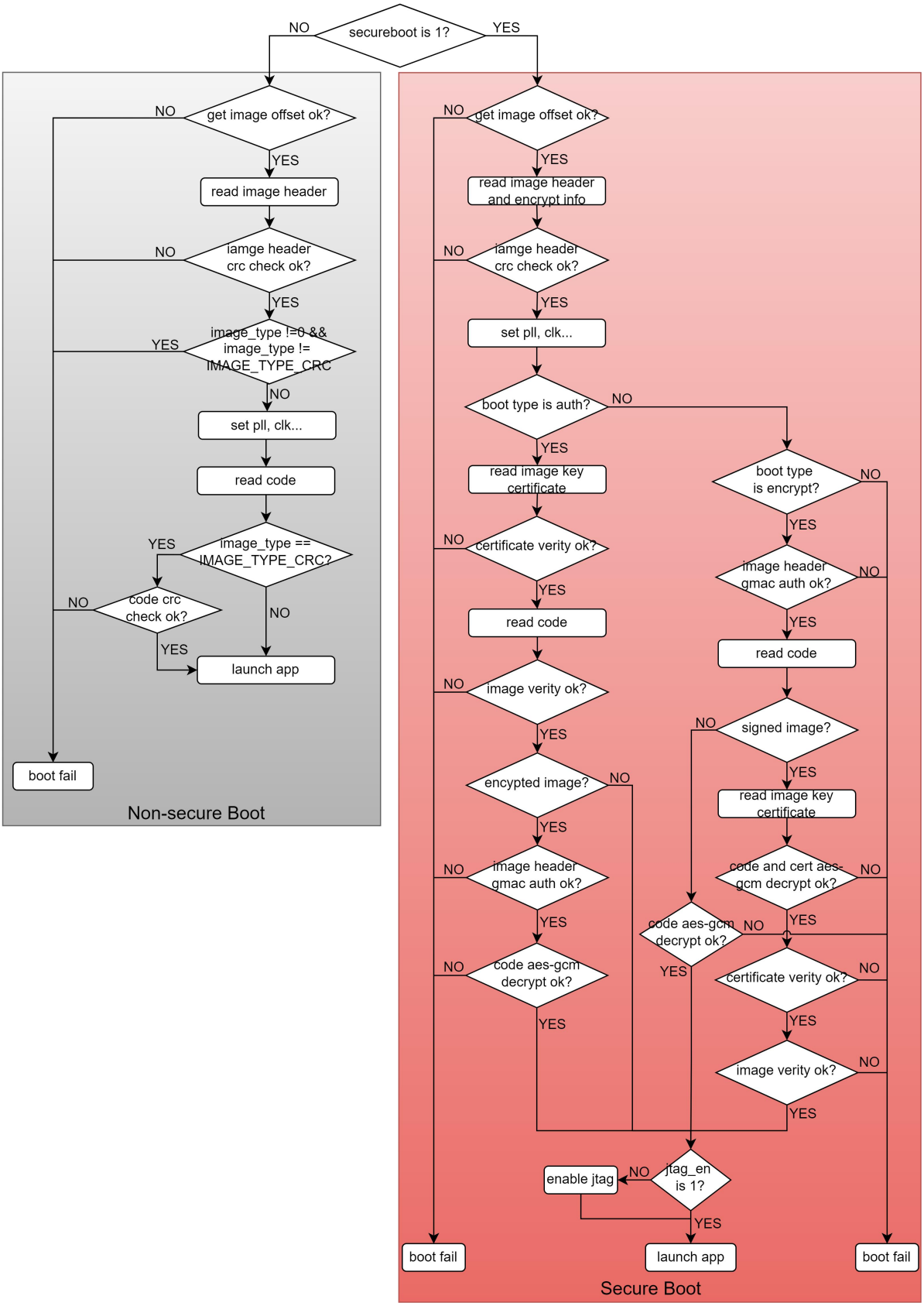
secureboot is 1?
NO
YES
get image offset ok?
read image header
iamge header crc check ok?
image_type !=0 && image_type != IMAGE_TYPE_CRC
set pll, clk...
read code
image_type == IMAGE_TYPE_CRC?
code crc check ok?
launch app
boot fail
Non-secure Boot
get image offset ok?
read image header and encrypt info
iamge header crc check ok?
set pll, clk...
boot type is auth?
read image key certificate
certificate verity ok?
read code
image verity ok?
encypted image?
image header gmac auth ok?
code aes-gcm decrypt ok?
boot type is encrypt?
image header gmac auth ok?
read code
signed image?
read image key certificate
code and cert aes-gcm decrypt ok?
code aes-gcm decrypt ok?
certificate verity ok?
image verity ok?
jtag_en is 1?
enable jtag
boot fail
launch app
boot fail
Secure Boot

图 1.2 安全和非安全启动流程图

1.3.1 RSA 签名认证

验证镜像的合法性的方法是验证镜像的 RSA 签名。工具生成镜像文件时如果选择了签名会使用 Imgk 私钥对原程序进行签名，Imgk 公钥则包含在“镜像密钥证书”当中。密钥证书本身会通过 Rotk 私钥进行签名，签名后的数据保存在证书最后。(证书数据格式详见表 1.1)

镜像烧写时，Rotk 公钥哈希值会烧录到 efuse。启动时签名认证首先会计算密钥证书里的 Rotk 公钥哈希值，之后和 efuse 保存的值进行对比，对比一致后使用该公钥来认证密钥证书合法性，认证合法后使用证书内的 Imgk 公钥来认证程序镜像合法性。(详细流程参考图 1.2)

Rotk 密钥支持更新，实现防回滚功能。由于 EFUSE 大小限制，只能支持四次更新，分别对应 CID 值 0x01,0x03,0x07,0x0f。镜像烧写时将新的 Rotk 公钥哈希值烧录到 efuse 对应位置，启动时会根据 CID 的值读取 efuse 对应位置的哈希值来和密钥证书里面的 Rotk 公钥计算得到的哈希值进行比较。

Offset	Size in bytes	Symbol	Description
0x00	1	Version	0x02
0x01	1	Certificate Identifier(CID)	CID field should be set the following 5 possible values only: 0x00, 0x01, 0x03, 0x07, 0x0F. This field should match bits[7:4] of EFUSE word 59.
0x02	2	Marker	0x4D43
0x04	260	Image public key	256-byte key modules followed by 4-byte exponent of the public key. Little-endian.
0x108	260	Root of Trust public key	256-byte key modules followed by 4-byte exponent of the public key. Little-endian. Note: SHA-256 (256-bit) hash little-endian of RoT public key is stored in EFUSE word 8~15.
0x20c	256	Image Key Certificate signature	RSASSA-PKCS1-v1_5 signature of Image Key certificate.

表 1.1 Image key certificate 表

1.3.2 AES-GCM 解密

工具生成镜像文件时如果选择了加密会使用密钥对镜像头信息和程序原文件进行加密。镜像烧写时，该密钥会烧录到 efuse 0 地址(该密钥仅被加密模块使用，其他模块均读不到该密钥值)。启动时会先计算头信息的 GMAC 值来和镜像中加密信息表(见表 1.2)的 Header TAG 进行对比，对比一致后使用加密信息表里面的 Image IV 来解密程序原文件并比较原程序的 TAG 值是否和镜像中的 Image TAG 一致(启动过程详见图 1.2)。

Offset	Size in bytes	Symbol	Description
0x00	16	Image IV	Image Initialization Vector (IV). A 128-bit user defined value used as initialization vector for AES128-GCM image decryption and tag verification.
0x10	16	Image TAG	AES-GCM Authentication tag of the encrypted image. Galois/Counter Mode (GCM) is a block cipher mode of operation that uses universal hashing over a binary Galois field to provide authenticated encryption. Authentication tag is computed for encrypted code area.
0x20	16	Header IV	Initialization Vector (IV). A 128-bit user defined value used as initialization vector for AES128-GCM header tag verification.
0x30	16	Header TAG	GMAC of the image header (0x0 – 0xF0) without this field. The 128-bit secret key programmed in EFUSE word 0~7 and Header IV in the header are used for computing the GMAC of header. GMAC is calculated using AES-GCM encrypt method. Only image header (excluding this field) is provided as 'Additional Authentication Data (AAD)' and plain text input as empty buffer. Header_tag = AES_GCM(Key = EFUSE_Key, Plain_text = 0, AAD = image[0 – 0xEF], IV = Header_IV);

表 1.2 Encrypt Info 表

1.4 镜像布局(Image Layout)

工具最终生成的镜像文件结构图如表 1.3，其中 Image Header 包含时钟配置、程序大小，加载位置等信息，之后是加密信息表，之后是程序原文件或者加密后的数据(选择了加密)，如果选择了签名的话之后是密钥证书和以上所有数据的签名数据。

Image header(192 bytes)
Encrypt Info(64 bytes) (no encypt image all data is 0xff)
APP Code
Image Key Certification(780 bytes) (only signed image)
Signature(256 bytes) (only signed image)

表 1.3 Image Layout 表

2. 程序镜像生成

首先使用 ImgcrGUI HV160 工具把 MCU 生成的 amt630hv160.bin 转 image.bin，点击 preheader 选择好 amt630hv160.bin 镜像文件，设置存储外设类型，并且点击 EMMC & PLL Config 配置 PLL 时钟（有默认值，不建议修改默认值）。最后保存输出文件即可。（流程请看图 2.1）

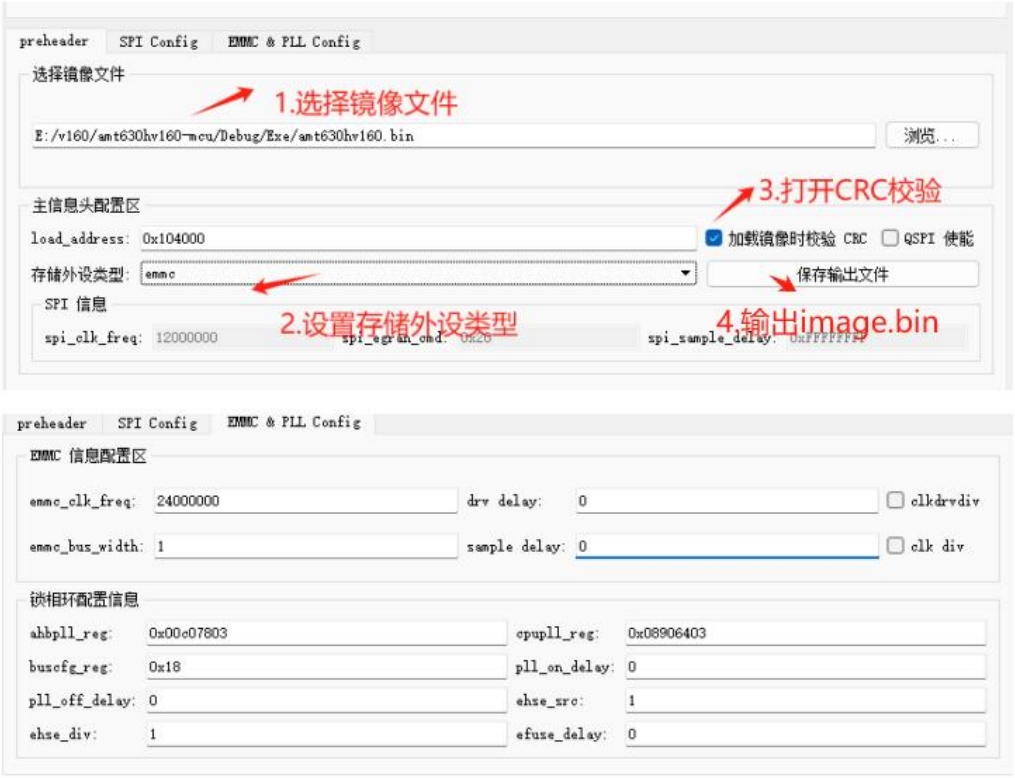

图 2.1 生成 image.bin 流程图

如果存储外设是 spinor 或者 spinand 又需要四线模式，则需要点击 QSPI 使能再点击 SPI Config 进行相应配置（图 2.2 是某个四线读命令时序所对应的配置）。另外若需要 bootrom 支持 norflash 四线模式启动的话需要 flash 本身 QE 位是使能的，即不需要额外的配置就可以直接识别四线读命令。

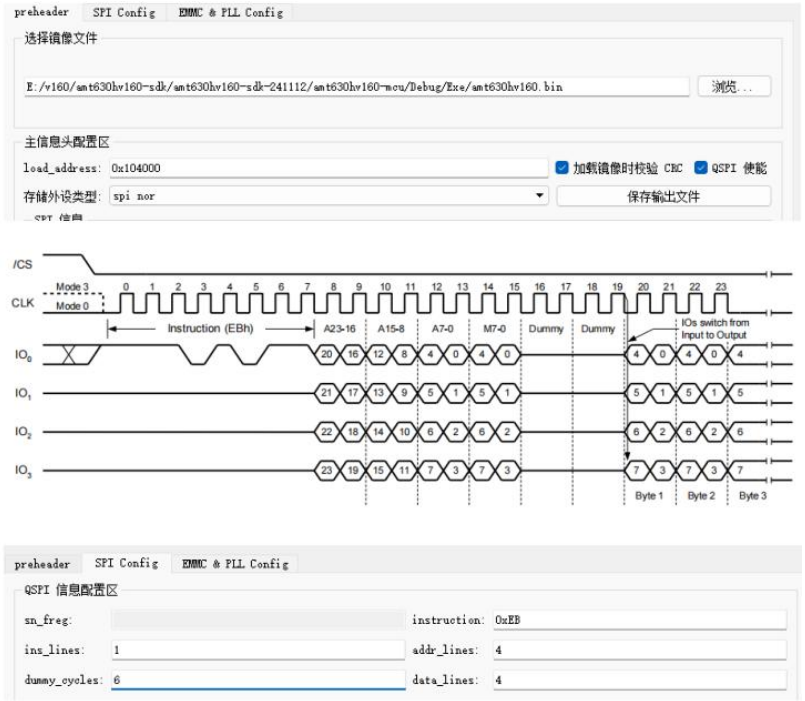

图 2.2 QSPI 读命令配置图

如果不需要安全启动的话直接升级该 image.bin 文件即可。

安全启动文件制作（流程请看图 2.10）。

第一步生成对称加密随机向量，用 genrandom 生成 iv.bin。Byte Count 输入 24 位点击 genrandom and save 保存名字为 iv.bin 文件，或者是直接点击 geniv and save 生成固定的 24 位 iv.bin 文件。

第二步生成非对称 RSA2048 加密密钥，用 genrsakey 生成 imgk.pem 和 rotk.pem 文件。Key Exponent value 一般选择默认 3。

第三步根据指定镜像密钥文件生成对应的私钥数据数组，提取 imgk.pem 对应的私钥文件（私钥自己保存）。

第四步生成对称加密随机密钥，Key Byte Count 可以选择 16 或者是 32 位，生成 aeskey.bin 和 aeskey.c 文件，其中 aeskey.c 文件中的数组要烧录到 efuse 中，需要替换 MCU 工程中 main.c 中的 aeskey 数组。

图 2.3 aeskey 数组

第五步根据指定的根密钥文件和镜像文件生成对应 ImageKeyCert 结构体内容，用 rotk.pem 和 imgk.pem 文件生成 cert.bin。

第六步根据指定的根密钥文件生成对应的公钥文件以及公钥的哈希数据文件，点击 genrotkpub，选择 rotk.pem 文件，并且选择 rotkpub.pem 和 rotkpub.hash 要存放的路径，点击生成得到的 rotkpub.hash 需要烧录到 efuse 中，所以需要替换掉 MCU 工程的 rotk_hash0 数组。

图 2.4 rotkpub_hash 数组

第七步加密镜像点击 encimage 添加 aeskey.bin，iv.bin，image.bin 生成 image_enc.bin 文件。

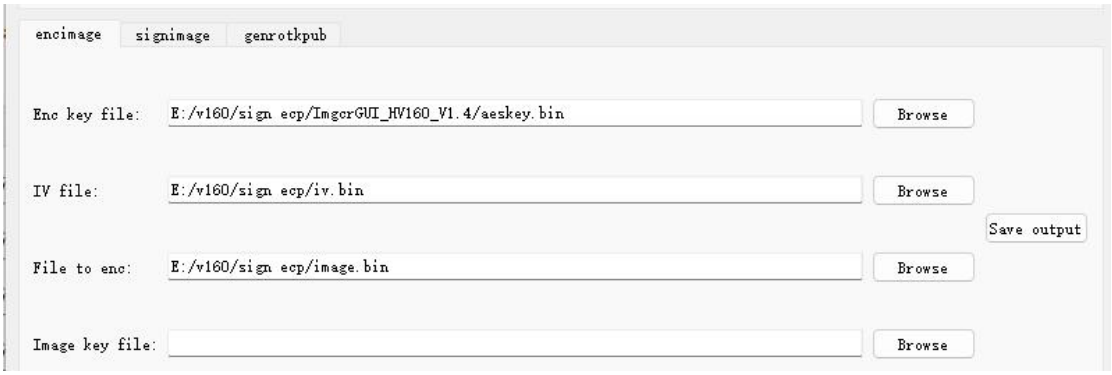

图 2.5 仅加密镜像图

第八步先加密后签名镜像，点击 signimage 添加 cert.bin，imgk.pem，image_enc.bin 与 aeskey.bin 生成 image_enc_sign.bin 文件。

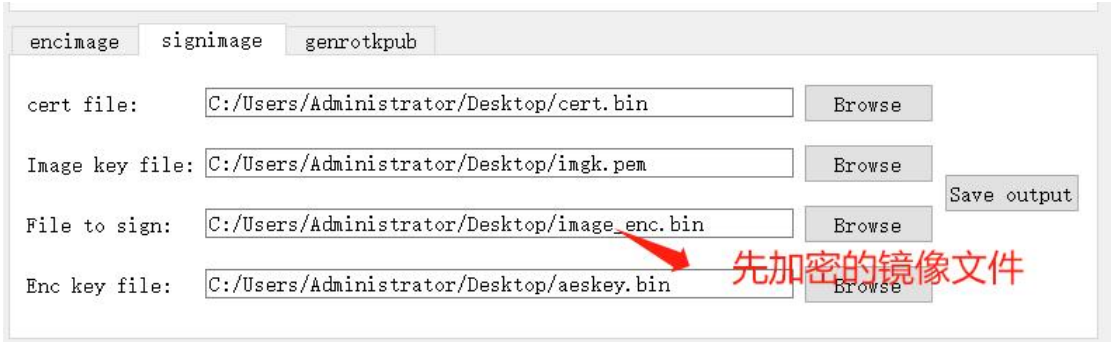

图 2.6 先加密后签名的镜像图

如果只签名的话在上面的基础删除第七步和第八步就可以，点击 signimage 添加 cert.bin，imgk.pem，image.bin 生成 image_sign.bin 文件。

图 2.7 仅签名镜像图

如果是先签名后加密的话，则先点击 signimage 添加 cert.bin，imgk.pem，image.bin 生成 image_sign.bin 文件，再点击 encimage 将 aeskey.bin，iv.bin，image_sign.bin 与 imgk.pem 生成 image_sign_enc.bin 文件。

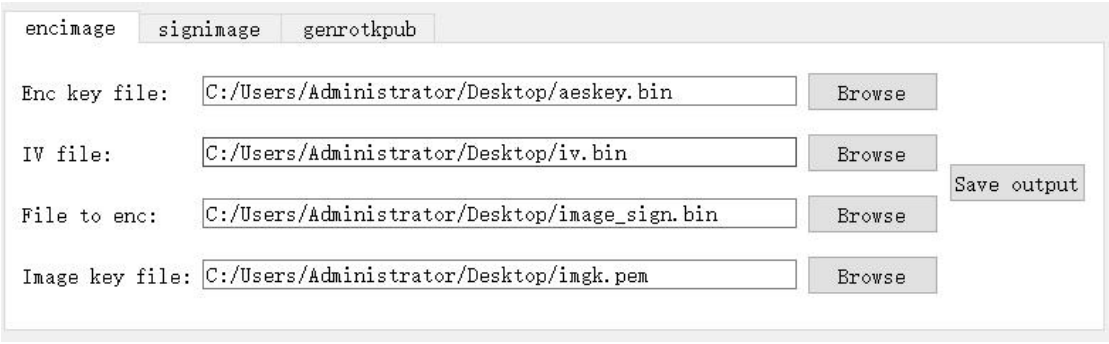

图 2.8 先签名后加密的镜像图

如果是仅仅加密的话则到第七步即可。

文件制作完成后需要修改代码，EFUSE_SECURE_BOOT_TYPE_AUTH 是代表镜像仅仅是签名了而已，EFUSE_SECURE_BOOT_TYPE_ENCRYPT 表示仅加密或者签名后再加密(通过 image_header 里面的 image_type 来区分)。

图 2.9 镜像加密签名写入 EFUSE_SCURUCE_BOOT_TYPE

图 2.10 镜像先加密后签名流程图